**Documentation Notes for GraphCreator**

Goal: Connect annotated word with Annotations of variable size at right border of text. (Both sides would allow for more annotations, but also increase the complexity of the problem a lot.)

How: Create Graph of all possible routes between words – nodes are located before and after each word in the space between lines, both above and below.

Issues: Automatic Linebreaks in Java don’t allow the user to retrieve the text of each line, leading to issues whenever a new line was triggered by something else than a space (‘ ‘), therefore the words are placed manually, and only spaces are considered.

Algorithm Idea: Greedy – Starting with the annotated word, route as far up and right as possible, to leave the most space for remaining annotations. Worst Case: Route diagonally across the whole page, using O(L+H) Nodes, where L = Line Width, and H = Page Height.

Problem: Algorithm breaks if Annotation can’t be placed above the annotated word – reverse Routing in this case?

This leads to more solvable positions with a very similar algorithm (If going up and left as much as possible isn’t sufficient, the target is unreachable anyways), but can trap an annotated word above the new Path, forcing an intersection, assuming the direction is evaluated on a node-by-node basis. (Idea: Using Annotation’s sizes, determine the area in which the routing will be reversed, then create all routes in reverse order, starting with the lowest annotation of the area)

One must also take precautions that no “regularly” routed node is routed above any previous node, as this too forces an intersection – this is easy to solve, as simply blocking the node of entering the area above and to the left of the last (few?) routed nodes is sufficient.

Random Notes:

Edges have a default weight of 1.0 at construction – other weights have to be set via the graph object

~~Allg. Routing ändern, dass es nicht sofort aufhört, wenn es den rechten Textrand erreicht!~~

~~Routen erst als “taken” markieren, wenn sie vollständig sind!~~ **🡨 Besseren Ort für Durchführung finden!**

~~Backtracking! (Tiefensuche) – immer nur Änderung, wenn vorher Pfad nach oben genommen wurde, sonst gibt es ohnehin keine Optionen mehr~~

Nicht in selbe Zeile wie “alte” Node kommen, sonst stößt man von hinten an 🡨 **Solved(Backtracking)**

~~Github-Account o.Ä.~~

~~Verbindungen nach oben zentrieren~~

~~Zusätzliche Verbindungen am/nach Zeilenende einbauen, vor allem bei unterschiedlich langen Zeilen!~~

~~Auch über mehrere Zeilen hinweg!~~

~~Verbindungne auch bei Leerzeichen von übermäßig langer Nachbarzeile?~~

**Dokumentation weiter schreiben!**

~~Puffer zwischen Annotationen und Textrand für zus. senkrechte Verbindungen – variable Größe! Evtl nicht als Graph sondern als “gesonderter Bereich” umsetzen? -> freiere Labelplatzierung, einfacher nachträglich verschiebbar (Siehe auch: OPO-Leader) 🡪 2. Durchang Routing (Weil sonst alle Pfadenden und Labelpositionen noch unbekannt)~~

~~OPO-Leader einbauen~~

Flussnetzwerk-Thesis lesen

~~Baccarbeit beginnen – Kann ich die auch auf Github hochladen? JA~~

**KIT hat Baccarbbeiten online! -> Zum Vergleich ansehen?**

**Kurven in Routingpfad für zu kleine Distanzen sehen hässlich aus! (Noch immer glitchy)**

**Graph darf nicht Knoten zu eng beienander haben! (Min= Kurvengröße?)**

**Paper schreiben – Routingalgorithmen genauer beschreiben!**

**Nachträglicher Pass: Routen begradigen? (🡪 Größere OPO-Leader) (Man kann immer bis 1 Zeile über Anfang des nächsten Labels gehen!)**

**Alternativ: #Kanten nach oben begrenzen 🡪 Geradere Routen!**

**Radikalbegradigung + OPO mit fixer Max-Kapazität? (🡨Also mehr oder weniger gerade zum Rand routen)**

~~Leader vorerst in fixe, vordefinierte Slots einordnen~~

~~Labels bekommen eigenen Zeilenabstand!~~

~~Dropdownbox o.Ä. zur Algorithmusauswahl!~~

~~Label-Abmessungen zugänglich machen~~ **(speichern?)**

~~Abmessungen in Routingalgorithmus einbeziehen!~~

~~Pseudokreuzungen vermeiden/graphisch besser darstellen!~~

~~ZUsammenstoßende Kurven eindeutig machen -> Ecke=Bezier-Kurve?~~

~~Eigene Klasse für Annotationen? (discuss!)~~

~~Routing-Algorithmus in externe Klasse verschieben!~~

~~Annotationen: /note{VolltextHier}~~

~~Bei Mehrfachverwendung von Kanten: Kreuzungsvermeidung ( und Sperre nach oben) verbessern!~~

Vergleich: Summe Labelhöhen (gezeichnet oder alle?) vs. Position in Zeile 🡪 Wissen, ob theoretisch genug Platz da ist 🡪 in einzelne Instanzen aufteilen!

//Reverse Routing: Bei “gefangener” Node Platz machen und altes Label am Rand weiter nach unten routen?

🡪Label sind nicht in Leserichtung angeordnet! (🡪Nach unten gehen verbieten, außer für letztes Label in Zeile?)

(Allg ähnliches Verfahren wie beim normalen Routing – nach unten so lange sperren, bis keine Kreuzung erzwungen wird)

Altenativ: Label so weit unten wie möglich, Algorithmus entsprechend angepasst

Beides auf einmal -> “Grenze” in der Mitte mit großem Konfliktpotential (Außerdem nicht so einheitlich)

Wenn Labelposition (oder zumindest –größe) bekannt, möglichst viel am Rand entlang gehen? (Zur Knickentfernung?)

Wie Kompromiss zwischen Knickentfernung und Überschneidungsfreiheit finden?

//Polygone aus höchster & niedrigster Route bilden -> Miteinander interferierende Punkte identifizierbar